U.S. Express Mail Label No. EL531624089US

PATENT
Docket No: 230074.0227

PATENT APPLICATION IN THE U.S. PATENT AND TRADEMARK OFFICE

for

LICENSE MANAGEMENT SYSTEM AND METHOD WITH MULTIPLE LICENSE SERVERS

by

Mark E. Redding, Logan A. Badia, Sandeep Handa, Hemant Sharma, Sanjay Chopra, Vikram Duvvoori, Shankar Ramamoorthy, and Ajay Tripathy

Cross-Reference to Related Applications

Embodiments of the present invention claim priority from Provisional Application Serial No. 60/152,785, filed September 3, 1999, and are related to U.S. utility patent applications entitled "License Management System And Method With License Balancing", Attorney Docket No. 230074.0228, filed ______; "System And Method For Selecting A Server In A Multiple Server License Management System", Attorney Docket No. 230074.0229, filed ______; and "License Management System And Method For Commuter Licensing", Attorney Docket No. 230074.0230, filed ______. The contents of each of these applications are incorporated by reference herein.

Background of the Invention

1. Field of the Invention

The present invention relates, generally, to license management systems and processes for managing licenses on a computer network and, in preferred embodiments, to such systems and processes involving a pool of license servers for managing software licenses among one or more users on the network.

2. Description of Related Art

The increasing popularity of computer usage in homes and businesses, as well as in governmental, research and education institutions, has brought about a world-wide demand for greater software variety and sophistication. Indeed, the software development industry in most industrialized countries has shown substantial growth in recent years and is expected to show continued growth through the next decade.

However, as software sophistication increases, development costs associated with such software also tend to increase. Modern software programs can require months or even years of development, often involving expensive resources and teams of highly skilled engineers and programmers, before a product may be readied for sale or license. Thus, software development companies are often required to make large investments early in the development of their products, in the hope that the products will provide a volume of sales or license revenues sufficient to cover their development investments and generate profits.

Illegal software usage and piracy have become a significant problem to software development companies. Because of the nature of computer software, illegal usage and illegal copying of proprietary software programs can be difficult to detect or deter. The increasing usage of computer networks has added to the problem. Computer networks can allow multiple users to access and copy software stored by a common network program server or copy and pass software between each other, over the network. A legitimately purchased or licensed copy of a software program available on a network could result in many illegitimate usages or copies by unauthorized or unlicensed users having access to the network.

Various forms of encryption techniques have been developed to inhibit usage of encrypted software by unauthorized users that do not possess a decryption program or key. However, such techniques typically require each authorized users to obtain or be passed a decryption program or key, in advance of usage of the encrypted program. Accordingly, such techniques can be prohibitively inconvenient for some computer and network environments, where it is difficult or impractical to supply each authorized user with a decryption program or key or to decrypt a program for each user or usage.

Other software protection techniques have required legitimate or authorized users to have a special hardware device or circuit installed in or plugged into their computer, prior to usage of the protected software. See, e.g., U.S. Patent No. 4,446,519 to Thomas. Again, such techniques can be prohibitively inconvenient for some computer and network environments. For example, if the network environment is such that the authorized user must use multiple computers on the network, then each computer must be provided with the special hardware device. Moreover, if the environment is such that computers having the special hardware device are not located in secure facilities, then unauthorized users may be able to access the protected software by using the non-secure computer in which the special hardware device is installed.

Accordingly, more sophisticated license management software has been developed for managing software licenses for computer networks, which do not require encryption of the protected software or special hardware devices in each authorized user's computer. For example, the assignee of the present invention, Rainbow Technologies, Inc., has marketed versions of a license management system under the trademark, SENTINELLM™. The SENTINELLM™ systems operate with a license server connected to a network of users. The license server stores and manages software licenses for the network users, in accordance with a license management program stored on the server. Each copy of a protected software program on the network is accompanied by a program code corresponding to a shell (also known as "wrapper") or library of Application Program Interface (API) functions, which communicates with the license management program on the server. When a user starts to run the protected software program, the shell code or library of API functions provided with the program communicates a request to use a license to the license server, over the network. The server, under the control of the license management software, responds to the request to determine whether it is storing an available license for the protected software program. If so, the server communicates an authorization message to the user and decrements a count of available licenses stored by the server. If not, the server communicates another message to the user, indicating that no licenses are available. In this manner, licenses are always stored and managed on a network license server. Each network user may have a copy of a protected software program, but must communicate with the license server for authorization to use the protected program.

License management systems which employ a license server, as discussed above, provide significant advantages with respect to the freedom to readily add, delete or change authorized users on the network, and the ability to readily control and alter licensing schemes (for example, to add or delete licenses or change conditions for licenses) by modifying only the license server files or programs. However, if all licenses are stored and managed in a single license server, failure of that server can result in a failure of the entire license management system. Accordingly, prior versions of the SENTINELLM™ systems include multiple license server capabilities, wherein two or more license servers are provided on the network, each having a pre-loaded license file and a license management program. One of the servers may be designated as a primary license server, while the others are designated as backup servers. If the primary server cannot be reached by a user, for example, because the primary server has crashed or otherwise gone down, the user may then communicate with a backup server to obtain an authorization message. The backup server, having a pre-loaded copy of the license file and the license management program, may then take over the license management functions.

Alternatively, in other prior versions of SENTINELLM™ systems, a shell program or library of API functions could be configured to send a general poll to all servers coupled to the communication channel on which the poll is sent. In response to a general poll, any server computer having a license file containing license information corresponding to the protected software program (whether or not the license information indicated that a license is available) would send a reply to the requesting client computer. The shell program or library of API functions associated with the requesting client computer would then respond to the first reply received from a license server having a license file storing license information for the protected software program. If the replying license server contains an available license for the protected software program, the replying license server provides an authorization message to the requesting client computer. If the replying license server does not contain an available license for the protected software program, the replying license server provides a message to the requesting client computer indicating that authorization to run the protected software program has not been granted. Such a system provides a degree of protection against system failure, in the event that one of the license servers crashes or goes down. The down server computer would not respond to the poll, but other server computers having appropriate license information in their license file would respond to the poll, thus allowing the client computer to continue to seek authorization to run the protected software program, even though one of the servers was down.

While the above SENTINELLM™ systems have operated well in many contexts, each license server operates somewhat independent of other license servers. Accordingly, there is a need in the industry for improvements in connection with management and coordination of multiple license servers (or a pool of license servers) in software license management systems for computer networks.

Summary of the Disclosure

Therefore, it is an advantage of embodiments of the present invention to provide a license management system and method for more efficiently managing licenses on a network using a pool of multiple license servers.

It is a further advantage of embodiments of the present invention to provide a system and method that facilitates communication between multiple license servers so that any license server in the pool may act as a leader server, should the leader server go down.

It is a further advantage of embodiments of the present invention to provide a system and method for managing licenses on a network using multiple license servers that allows a client to locate and receive an authorization from a new license server, should the licenser server that previously issued the authorization go down.

It is a further advantage of embodiments of the present invention to provide a system and method for managing licenses on a network using multiple license servers that allows allocations for multiple vendors to be managed with a single process running on a single license server.

It is a further advantage of embodiments of the present invention to provide a system and method for managing licenses on a network using multiple license servers that allows a redundant license file containing a new hard limit of allocations and a new initial distribution of allocations to be loaded onto one license server and then copied to other license servers in the server pool.

These and other advantages are accomplished according to a system for managing licenses for protected software on a communication network. The system comprises at least one client computer and a pool of license servers coupled to the communication network. The client computers request authorizations to use the protected software. The pool of license servers is programmed for managing licenses to use the protected software.

Within the pool of license servers is a current leader server, programmed for managing all license servers in the pool, and at least one follower server programmed for managing the distribution of authorizations allocated to that particular follower server. Each license server stores a status of the authorizations allocated to that particular license server, and each follower server communicates the status of the authorizations allocated to that particular follower server to the current leader server. Each follower server is capable of becoming a new leader server if the current leader server can no longer manage the distribution of authorizations for the license servers.

These and other objects, features, and advantages of embodiments of the invention will be apparent to those skilled in the art from the following detailed description of embodiments of the invention, when read with the drawings and appended claims.

Brief Description of the Drawings

FIG. 1 is a generalized block diagram representation of an example network environment according to an embodiment of the present invention.

FIG. 2 is a generalized representation of a redundant license file (RLF) for the network environment of FIG. 1 according to an embodiment of the present invention.

FIG. 3 is a generalized representation of a license code contained in an RLF of FIG. 2 according to an embodiment of the present invention.

FIG. 4 is a generalized representation of a license code contained in an RLF and copied into a license table and a distribution table within a single license server for the network environment of FIG. 1 according to an embodiment of the present invention.

FIG. 5 is a generalized representation of an initial state of the distribution tables of leader server A and follower servers B and C in a server pool comprising three license servers according to an embodiment of the present invention.

FIG. 6 is a generalized representation of the distribution tables of leader server A and follower servers B and C in a server pool comprising three license servers in the example of FIG. 5 after an authorization has been granted by follower server B according to an embodiment of the present invention.

FIG. 7 is a generalized representation of the distribution tables of leader server A and follower servers B and C in a server pool comprising three license servers in the example of FIG. 6 after an authorization has been granted by follower server C according to an embodiment of the present invention.

FIG. 8 is a generalized representation of the distribution tables of leader server A and follower servers B and C in a server pool comprising three license servers in the example of FIG. 7 after an authorization has been granted by leader server A according to an embodiment of the present invention.

FIG. 9 is a generalized representation of the distribution tables of license servers A, B and C in a server pool comprising three license servers in the example of FIG. 8 after old leader server A has gone down and follower server B has been elected as the new leader server according to an embodiment of the present invention.

FIG. 10 is a generalized representation of the distribution tables of license servers A, B and C in a server pool comprising three license servers in the example of FIG. 9 after a client who formerly received an authorization from old leader server A has now received an authorization from new leader server B according to an embodiment of the present invention.

FIG. 11 is a generalized representation of the distribution tables of license servers A, B and C in a server pool comprising three license servers in the example of FIG. 10

after old leader server A has come back on line as a follower server according to an embodiment of the present invention.

Detailed Description of Preferred Embodiments

In the following description of preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the preferred embodiments of the present invention.

Preferred embodiments of the invention relate to a system and process involving a pool of license servers for managing licenses to, for example, one or more protected software programs, files or other data structures, among one or more users on the network. Protected software may include, but is not limited to, for example, a software program, such as a word-processing program, a graphics program, a computer game, etc., a proprietary file or other data structure, such as a data-base or other form of data, as well as other software encoded information or instructions, for which the control of user access is desired. For purposes of simplifying the present disclosure, the protected software used in the following examples is one or more proprietary software programs.

According to a preferred embodiment of the present invention, the plurality of license servers are managed in accordance with a server pool scheme, as controlled by a license management program associated with each license server computer and the shell program or library of API functions associated with each copy of the protected software program. In preferred embodiments, the pool of license servers comprises three to 11 servers. However, it should be noted that alternative embodiments of the present invention are not limited to any particular maximum number of license servers. In addition, in further alternative embodiments, multiple pools of license servers can reside on a single network. However, for purposes of illustration only, embodiments of the present invention will be described herein with reference to a pool of three license servers.

An example hardware environment for an embodiment of the present invention is illustrated, generally, in FIG. 1. With reference to FIG. 1, a computer network 10 includes a plurality of user or client computers 12 and three license servers 14, each coupled for communication over a communication network link, generally referenced at 16. The plurality of client computers 12 are identified as "Client 1", "Client 2", and "Client N", and the plurality of license servers 14 are labeled as "Lic. Server A", "Lic. Server B", and "Lic. Server C." Embodiments may employ any suitable number of client computers 12 and any suitable number of license servers 14. Also, while not shown in FIG. 1, the network 10 may include additional components, including one or more program or file servers, routers and/or other well known network devices and resources.

Each client computer 12 preferably includes a suitable processor and associated transient memory, such as an RAM, for running a protected software program. The client computer may be part of a standard personal computer (PC), network terminal, workstation or the like. In one preferred embodiment, each client computer 12 is coupled to a persistent program storage memory device 18, which may include, but is not limited to, a hard disc drive, floppy disc drive, tape drive, CD-ROM or the like, having a computer readable medium on which the protected software program is stored. Also stored as part of the protected software program is additional program code, such as code corresponding to a shell or library of API functions as discussed above, for communicating with the license servers which are under control of a license management program, in accordance with communication functions discussed below.

Each license server 14 preferably includes a suitable processor and associated transient memory, such as an RAM, for running a license management program as described herein. In addition, each license server 14 is coupled to one or more persistent program storage memory devices 20, which may include, but is not limited to, a hard disc drive, floppy disc drive, tape drive, CD-ROM or the like, having a computer readable medium on which a license management program 22 and a redundant license file (RLF) 24 are stored.

An example embodiment of an RLF 24 is shown in FIG. 2 as containing license codes 26 for proprietary software programs A, B, C, etc. Each license code 26, in one example embodiment, comprises a string of data relating to license policy and the software program to which the license policy applies. In preferred embodiments, license codes 26 for software produced by more than one vendor can reside within the same RLF 24 of the same license server 14, and may be managed with a single process (execution of the license management software) running on a single license server 14. Thus, any given licenser server 14 does not need to have multiple license management processes running to manage licenses for multiple protected vendor applications. Instead, only one license management program needs to be running on a given license server 14, to manage licenses for protected vendor applications managed by that license server.

In the example embodiment of FIG. 3, a license code 26 comprises a data string defining multiple records or fields R1, R2, R3 ... RN, wherein each record corresponds to an attribute associated with the license policy, the software program to which the license policy applies, or other information. In preferred embodiments, each license code 26 includes at least one attribute associated with a license policy, the number of allocations for using the protected software program, and at least one other attribute associated with the identity of the protected software program. The number of allocations for using the protected software program is the maximum number of users that can be running the protected software program at any one time, and is also referred to as the ceiling or hard limit. Data associated with a license policy preferably includes data representing the expiration date or expiration time of the license, as granted by the licensee (e.g., the protected software program's owner or developer).

For purposes of illustration only, in the embodiment of FIG. 4 only one fictional license code 26, identified as "Application v1.0," is stored in the RLF 24 of a license server 14, with a hard limit of 100 total allocations distributable across all license servers 14 (see reference character 28). In preferred embodiments, another attribute of the RLF 24 is the IP address 32 for each license server 26. Another attribute of the RLF 24 is the distribution of the 100 allocations across all the license servers 14. This distribution is identified as the initial distribution 30, and is configurable by the network administrator. In the example of FIG. 4, an initial distribution 30 of A:40, B:30, C:30, indicates that 40 allocations will initially be assigned to license server A, 30 allocations will initially be assigned to license server B, and

30 allocations will initially be assigned to license server C. This initial distribution 30 also signifies that the pool will comprise three license servers. If the network administrator makes an initial distribution that does not equal the hard limit of allocations (see reference character 28), in preferred embodiments of the present invention the extra allocations will be put into a free pool 48 maintained within the distribution table 36. If, on the other hand, the network administrator makes no initial distribution 30 of allocations in the RLF 24, preferred embodiments will divide the allocations evenly over the number of servers in the pool, and any extras will be put in the free pool 48. For example (but not illustrated in FIG. 4), if no initial distribution 30 was assigned by the network administrator, information representing an even distribution of A:33, B:33, and C:33 would be stored in the initial distribution attribute 30 of the RLF 24, and when a particular license server 14 is started, the RLF 24 would load an even distribution of A:33, B:33, and C:33 into the current distribution attribute 40 of distribution table 36, and would load a value of one into the free pool 48.

In preferred embodiments of the present invention, each RLF 24 stored in each license server 14 is an exact copy of every other RLF 24 of license servers in the pool. Thus, attributes of the license code 26 such as the expiration date or number of allocations need not be passed between license servers 14 in response to a request for authorization to use a protected software program from a client computer 12, because an exact copy of the license code 26 has already been stored on the hard disk of each license server 14 prior to the startup of that license server 14.

Each license server 14 operates, under the control of its associated license management program 22, to perform license management functions in association with data contained in the RLF 24, as described herein. Thus, when a particular license server 14 is started, the license server 14 loads the contents of its RLF 24 into a license table 34 in RAM or other memory and reads the license table 34, which identifies that server as a license server 14. It should be noted that in preferred embodiments, the license table 34 is never modified. In addition, the contents of the RLF 24 is loaded into a distribution table 36 in RAM or other memory, and the initial distribution 30 is further copied into another record, distinct from the license code 26, identified as a current distribution 40. Unlike the license table 34, the distribution table 36 changes its current distribution 40 over time to keep track of current allocations.

In preferred embodiments of the present invention, when the license servers 14 in the server pool are first started up, the first license server 14 to come on line is designated as a leader server. The leader server always has a global picture of the current distribution of allocations within every license server 14 in the server pool. If none of the license servers 14 were definitively started first, the license server 14 having a higher priority as established by a leader priority list will be designated the leader server. The leader priority list is stored in the RLF 24 and is read at the time of server startup. This is known as a leader election process. All other license servers 14 are designated as follower servers.

The ability to intentionally start up a particular license server 14 and force it to be the leader server is advantageous because it may be desirable for a particular license server 14 on a particular subnet to operate as the leader server due to its physical location, speed, or efficiency. In prior art systems, the leader server always had the highest IP address, and to change leader servers required physically reconnecting to a different address or reconfiguring the license servers by changing the IP address. In embodiments of the present invention, however, the leader server may be changed simply by turning a particular license server 14 on first.

For purposes of illustration only, FIG. 5 illustrates an example of the distribution tables 36 for a three server pool consisting of license servers A, B, and C. Assume, for this illustration, that all three license servers have been started up, and that license server A has been designated as the leader server, and B and C as the follower servers. Further assume that the hard limit of allocations for the software program Application v1.0 is 100, as indicated by the hard limit record 28 associated with the license code 26 for Application v1.0 within each distribution table 36. In addition, assume that the current distribution of allocations is 40 on leader server A, 30 on follower server B, and 30 on follower server C, as indicated in the current distribution record 40 associated with the license code 26 for Application v1.0 within each distribution table 36.

Note also that in the embodiment of FIG. 5, associated with each license code 26 is a record for available allocations for each server in the pool (see reference character 38), a record for available allocations for all servers in the pool (see reference character 44), and a record for allocations currently in use for each server in the pool (see reference character 42). The allocations in these records are values which are incremented or decremented as authorizations are issued, returned, or borrowed. It should be understood that the records identified by reference characters 28, 38, 40, 42, and 44 in FIG. 5 are associated with a particular license code 26, but are distinct from it.

As indicated in FIG. 5, in preferred embodiments of the present invention the structure of the distribution tables 36 will be the same for both the leader server A and follower servers B and C, but only the distribution table 36 for leader server A (the leader distribution table) will reflect the allocation status of the other license servers 14 in the server pool. In contrast, the distribution tables 36 for follower servers B and C (the follower distribution tables) will only reflect the allocation status of that particular follower server, as indicated by the X (don't care) designations in portions of the follower distribution tables.

When a user at a client computer 12 desires to run Application v1.0 from a follower server, such as follower server B, for example, the client computer 12 may first load some or all of the protected program into the transient memory of the client computer 12, along with the program code corresponding to a shell or library of API functions. Selection of a follower server from which to request authorization is described in a related U.S. utility application entitled "System and Method for Selecting a Server in a Multiple License Server License Management System," attorney docket no. 230074/0229, filed ________, the contents of which are incorporated by reference herein. Alternatively, the protected program may remain in persistent memory 18 until and unless the server computer communicates an authorization signal to the shell program or library of API functions. Loading of the shell program or library of API functions is preferably transparent to the user on the client computer 12 and, preferably, occurs in response to the user inputting a command to open the protected software (for example, by clicking a mouse button on an icon associated with the protected software).

As part of the function of the shell or library of API functions, a request is then sent from the client computer 12 to follower server B for one or more authorizations to run the protected program. For purposes of this example, assume that client computer 12 requested only one authorization. Follower server B, under the control of the license management software, responds to the request by looking at its distribution table 36 to determine whether it has available allocations for Application v1.0. In the example of FIG. 5, follower server B has 30 allocations available, as represented by the allocations available record for each server in the pool (reference character 38) in the distribution table 36 for follower server B. Because it has available allocations, follower server B communicates an authorization message to the client computer 12. As illustrated in FIG. 6, once the authorization message is sent, the allocations available record 38 for follower server B decreases to 29, and the allocations in use record 42 for follower server B increases to one. If an insufficient number of allocations are available to completely satisfy the request, no authorizations are sent, and follower server B communicates another message back to the client computer 12, denying the request. In preferred embodiments, follower server B will then issue a borrow request in an attempt to borrow available allocations from another source. Borrowing is described in a related U.S. utility application entitled "License Management System and Method with License Balancing," attorney docket no. 230074/0228, filed ________, the contents of which are incorporated by reference herein.

As noted above, in preferred embodiments of the present invention the leader server always has a global picture of the distribution of allocations in the whole server pool. Thus, any changes to the distribution table of follower server B must be communicated to leader server A. In preferred embodiments, follower server B can determine the IP address for the leader server A from a leader priority list 46, which is configurable by the network administrator. It should be noted that in preferred embodiments, the IP address of license servers in the server pool are stored in the leader priority list 46, a data structure separate from the license codes 26, to facilitate faster lookup. However, in alternative embodiments the IP addresses may be stored in a record associated with the license codes 26. In any case, after determining the IP address of the leader server A, follower server B communicates the fact that its available allocations count has dropped to 29 to leader server A, and the distribution table of leader server A will be updated accordingly. Thus, as illustrated in FIG. 6, the distribution table 36 of leader server A reflects that the allocations available record 38 for follower server B has dropped to 29, the allocations in use record 42 for follower server B has increased to 1, and that the record for allocations available for all servers in the pool (see reference character 44) has dropped to 99.

Similarly, if another client computer 12 requests authorization to run Application v1.0 from follower server C, follower server C will look at its distribution table 36, determine that it has 30 allocations available, and send an authorization back to the client computer 12. As illustrated in FIG. 7, once the authorization is sent, the allocations available record 38 for follower server C drops to 29, and the allocations in use record 42 for follower server C increases to one. This change is then communicated to leader server A, whose distribution table 36 will reflect that the allocations available record 38 for follower server C has dropped to 29, the allocations in use record 42 for follower server C has increased to one, and that the record for available allocations for all servers in the pool (see reference character 44) has dropped to 98. It should be noted in FIG. 7 that the distribution tables 36 for follower servers B and C keep track of changes to their own allocations, but do not keep track of changes to the allocations of other license servers.

Continuing the example of FIG. 7 for purposes of illustration only, assume that a client computer A has requested and received an authorization to run Application v1.0 from leader server A. As illustrated in FIG. 8, once the authorization is sent, the allocations available record 38 for leader server A drops to 39, the allocations in use record 42 for leader server A increases to one, and that the record for available allocations for all servers in the pool (see reference character 44) has dropped to 97. Thus, at this point in time, one client computer 12 is connected with each license server 14.

In embodiments of the present invention, the license servers 14 periodically communicate with (ping) each other so that the license servers 14 know which other license servers 14 are down or up. The communication, or pinging, may be in the form of a periodic signal (heartbeat) sent from the leader server to the follower servers. Server identification information, such as the IP address of the follower server, may be contained within this heartbeat, but license code information need not be communicated. If the leader server does not receive an acknowledgement (ack) from a particular follower server in response to the heartbeat within an expected period of time, the leader server will designate that follower server as being down. In addition, if a follower server does not receive a heartbeat from the leader server within an expected period of time, it sends a leader check message to the leader server to check if the leader server is still alive. If the leader server is alive, it will send an "ack" in response to the leader check message. If no "ack" is received, the follower server will start a leader election process to select a new leader server. Selection of a new leader server is described in a related U.S. utility application entitled "System and Method for Selecting a Server in a Multiple License Server License Management System," attorney docket no. 230074/0229, filed ________, the contents of which are incorporated by reference herein.

In communications from leader to follower servers, each message contains updated server status (up/down) for each server in the pool. When a follower server receives a message from the leader server, the follower server updates its server status information, which is used when initiating the leader election process. It should be understood that follower servers do not ping each other to check if each other is alive, because that would generate unnecessary network traffic. Thus, the follower servers rely on status information passed to them by the leader server.

In preferred embodiments of the present invention, if the leader server sends acknowledgement heartbeats to the followers, and one of the follower servers (B, for example) receives the acknowledgement but follower server C does not, only follower server C will send one or more re-try heartbeats. If follower server C still does not receive an acknowledgement to the re-try heartbeats, follower server C will communicate with follower server B and attempt to initiate a process to select a new leader server. However, follower server B has already received an acknowledgement from leader server A, so follower server B will communicate with leader server A and inform leader server A that follower server C is initiating a process to select a new leader server. At this time, leader server A will send a new heartbeat to follower server C. If leader server A fails to receive a reply heartbeat from follower server C, and follower server C again contacts follower server B about a process to select a new leader, then follower server B will communicate to follower server C that a leader server already exists in the server pool. At that point, follower server C will disassociate itself from the pool.

Now assume that leader server A goes down. Because the license servers 14 are periodically pinging each other, the two follower servers B and C will soon determine that leader server A has gone down. Further assume that follower server B is then selected as the new leader server. New leader server B will already have its own RLF 24, license table 34, and distribution table 36, which includes the current distribution of allocations 40. However, new leader server B does not have the global allocation information maintained by the old leader server A. Therefore, upon becoming the new leader server, license server B must now receive this information from the other license servers. As illustrated in FIG. 9, because license server A is now down, within the distribution table 36 for new leader server B, the allocations in use record for license server A (see reference character 42) is set to zero, and the allocations available record (see reference character 38) is set to 40. When new leader server B sends a heartbeat to follower server C, the heartbeat contains information which identifies B as the leader server. Follower server C will send an "ack" in response to the heartbeat from new leader server B, and in the process, will provide information on the allocations in use record and allocations available record for follower server C. Once this information is received, within the distribution table 36 for new leader server B, the allocations in use record for license server C (see reference character 42) is set to one, and the allocations available record (see reference character 38) is set to 29. In addition, the record for available allocations for all servers in the pool (see reference character 44) is set to 98. In this manner, new leader server B receives a global picture of the status of the license servers in the server pool.

Meanwhile, because client computers 12 connected to a particular license server 14 are also periodically communicating with each other, for example, using a regular heartbeat function that indicates that each is still functioning, client computer A will soon determine that server A went down. Client computer A knows that it has received an authorization to run the protected program from server A, and therefore knows that it must enter a fail-over mode and look for a new license server 14 from which to receive an authorization. Because client computer A receives leader priority list 46 from a license server 14 whenever client computer A receives an authorization, client computer A knows the IP address of every license server 14 in the server pool. Client computer A then sends a heartbeat to the remaining license servers 14 in turn, in the order designated in leader priority list 46, until it finds a server (B, in this example) with an available allocation.

In preferred embodiments of the present invention, each license server 14 maintains a key table containing a list of client computers 12 that currently have authorizations issued by that license server 14. Thus, when new leader server B receives a heartbeat from client computer A, new leader server B will determine from the key table that it did not previously issue an authorization to client computer A. Thus, new leader server B will convert the heartbeat to a request, and will reserve the requested number of allocations for client computer A, provided it has the requested number of allocations available. In this manner, as client computers 12 formerly connected to old leader server A locate and ping new leader server B, new leader server B will issue authorizations and accept the allocations of old leader server A. The distribution table 36 of new leader server B will then change accordingly. As illustrated in FIG. 10, once client computer A is authorized by new leader server B, the distribution table for new leader server B will be updated such that the allocations in use record (see reference character 42) for leader server B is increased to two, and the allocations available record (see reference character 38) is decreased to 28.

If client computer A, in fail-over mode, sends a heartbeat to new leader server B, but new leader server B doesn't have an available allocation, then new leader server B will send a denial message to client computer A. It should be noted that new leader server B may also seek to re-allocate or borrow an allocation from another license server. Re-allocation and borrowing is described in a related U.S. utility application entitled "License Management System and Method with License Balancing," attorney docket no. 230074/0228, filed ________, the contents of which are incorporated by reference herein. In response to the denial message, client computer A will send its next heartbeat to the next license server 14 in the key table (follower server C in this example). If follower server C has an available allocation, it will convert the heartbeat to a request, and issue an authorization to client computer A.

If old leader server A should come back up, it will be a follower server, and have an initial distribution according to the allocations for old leader server A in the distribution table of new leader server B, regardless of what is contained in the RLF 24. Old leader server A can become the leader server again, if new leader server B goes down. When old leader server A comes back up as follower server A, follower server A sends a heartbeat to new leader server B, who then recognizes that follower server A is back up. New leader server B then transmits the current allocation of old leader server A to follower server A. Continuing the example of FIG. 10 for purposes of illustration only, if old leader server A comes back up as follower server A, the current allocation of old leader server A (see reference characters 42 and 38) in the distribution table 26 of new leader server B, which is zero allocations in use, 40 available, will be communicated to follower server A. This allocation is reflected in the distribution table 26 of follower server A in FIG. 11.

In alternative embodiments of the present invention, before bringing old leader server A back up as follower server A, the network administrator may use this opportunity to modify the RLF 24. Modification of the RLF 24 may be desirable because a new license code 26 was added, or because a modification of the initial distribution of allocations 30 for a given license code 26 already in the RLF 24 is desired.

In preferred embodiments of the present invention, each RLF 24 has an associated sequence or version number. Each time the RLF 24 is changed, the sequence number is incremented. Thus, if license servers should have RLFs 24 different sequence numbers, some RLFs 24 are not up to date. Continuing the example above for purposes of illustration only, when old leader server A comes back up as follower server A with a new RLF 24, it sends a "server up" message which includes the sequence number of its RLF 24 to the other license servers 14.

When the other license servers B and C respond to follower server A's "server up" message, license servers B and C will send a reply back to follower server A which includes their own sequence numbers. When follower server A receives these replies, it compares the sequence numbers. In the present example, follower server A will determine that it has the highest sequence number, indicating that follower server A has the latest RLF 24. Follower server A will then send a message to new leader server B indicating that follower server A has the latest RLF 24. New leader server B will then acknowledge this message and goes into a receive mode, at which time follower server A copies the RLF 24 to new leader server B. Once new leader server B has received a copy of the latest RLF 24, new leader server B notifies other functioning follower servers that it has a new RLF 24, and the new RLF 24 is then copied to the other functioning follower servers. In the preferred embodiment described above, the leader server is updated with the new RLF 24 before any other license server 14 in the pool, because, at any point in time, the leader server should have the correct and the latest information. After the license servers 14 receive the new RLF 24, their license tables 34 and distribution tables 36 are updated based on the new RLF 24.

The process described above is advantageous in that it allows a network administrator to update the RLF 24 on each license server 14 without taking each license server 14 off line, loading the new RLF 24, and then bringing it back up. Instead, the RLF 14 for license servers 14 in the server pool can be updated by bringing only one license server 14 down.

In alternative embodiments of the present invention, the RLF 24 may be updated without the necessity for bringing a license server 14 down. In such embodiments, the network administrator would initiate a utility from a client computer 12 that sends an "update RLF" message containing a new RLF 24 to a particular license server 14. The particular license server 14 would then proceed as described above.

It should be noted that in embodiments of the present invention, a network administrator can also use this process to add a license server to the pool by simply bringing the new license server 14 on line. When the new license server is brought on line, it may have a new RLF 24.

In addition to the purposes described above, the heartbeats communicated between the follower servers and the leader server are also used to enforce majority rule. Majority rule is a security measure to prevent excessive authorizations from being issued.

License servers 14 operating under majority rule will only issue an authorization if a majority of license servers 14 in the server pool are up and running. Continuing the example server pool described herein for purpose of illustration only, suppose the two follower servers B and C go down. Because of the communication between license servers 14, leader server A will know that only one license server – itself -- is still operating. Leader server A will also check a key file, which stores the total number of license servers 14 in the server pool, and note that the key file stores a value of three. Although leader server A is still up and running, because there is no majority of license servers running (only one of the three license servers are running), leader server A is precluded by the majority rule from issuing authorizations.

Without the majority rule, which may be ignored in alternative embodiments of the present invention, the network could be intentionally segmented such that each license server 14 becomes isolated from every other license server 14. With the license servers 14 isolated, each license server 14 would appear to be the only license server 14 in the server pool, and thus would operate as a leader server. It would then be possible to receive authorizations from each of these independent leader servers for all allocations, for a total of three times as many allocations for each legitimate license.

Therefore, embodiments of the present invention provide an improved system and method for managing licenses on a network using license servers that cooperate with one another and efficiently manage allocations in a network environment. The improved system and method facilitates communication between license servers so that any license server may act as a leader server should the leader server go down. In addition, embodiments of the present invention allow a client to locate and request an authorization from a new license server, should the licenser server that previously issued the authorization go down. Furthermore, with embodiments of the present invention, allocations for multiple vendors can be managed with a single process running on a single license server. Embodiments of the present invention also allow am RLF containing a new hard limit of allocations and a new initial distribution of allocations to be loaded onto one license server and then transferred to other license servers in the server pool.